SIGNAL REPRESENTATION AND DATA COMPRESSION USING PARTIAL REALIZATIONS

by

EYYUNNI VENUGOPAL

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

AUGUST, 1991

SIGNAL REPRESENTATION AND DATA COMPRESSION USING PARTIAL REALIZATIONS

A Thesis Submitted

in Partial Fulfilment of the Requirements

for the Degree of

Master of Technology

by

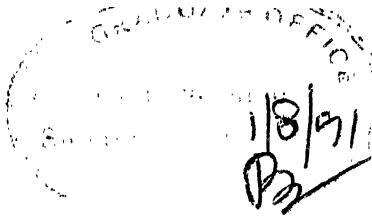
EYYUNNI VENUGOPAL

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

AUGUST, 1991



CERTIFICATE

It is certified that the work contained in the thesis entitled "**Signal Representation and Data Compression using Partial Realizations**", by Eyyunni Venugopal, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

S.K. Mullick

Professor

Department of Electrical Engineering

I.I.T. Kanpur

ABSTRACT

Partial realizations of sequences over a finite field are obtained, using Berlekamp-Massey algorithm in both one dimension and two dimensions. This representation is applied to compress the image data. It has been established, that, through deterministic modelling of a stochastic behaviour, data compression cannot be achieved. However, by employing a fidelity criterion it was shown that some compression can be achieved. The introduction of error bound to two dimensional sequences could not be done because of some implementation problems. To get an optimal representation in the one dimensional case, the problem was shown to be NP-hard.

To

My Parents and Teachers

ACKNOWLEDGEMENTS

I am grateful to my thesis supervisor, Dr. S.K. Mullick, for suggesting the topic for study. His constant encouragement, inspiration and constructive criticism helped a long way in shaping the thesis to the present form.

I thank *Motorola Inc.*, for providing me financial assistance.

The stimulating discussions with Madhu and Udaya are gratefully acknowledged.

I wish to thank Ramprasad, for helping me in taking the hard copies of the images.

I thank Ravi, Perraju and Ramudu, for helping me in the preparation of the thesis.

There are any number of people who have been supportive throughout my stay at IITK. Particular mention goes to Hari, Babu, Srikanth, Murali, Reddy, Kalari, Das, Sandy, Arun.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Source coding as opposed to error control coding aims at removing the redundancy in the data . The present day source coding techniques may be classified into two broad catagories. One is source coding with error and the other one being distortionless coding. Linear predictive coding and transform coding come into the former category [13] . In these techniques for fixed compression ratios, the distortion is minimized by taking certain norm which is the mean square error. Run length coding and Ziv-Lempel coding come into the latter class [11,24].

Source coding techniques have their theoretical basis in information theory. A mathematical treatment of the measure of information was given by Shannon. His approach is what is presently known as the classical information theory. In 1965, yet another concept of information came into picture, based on the complexity theory. Chaitin and Kolmogorov [5,6,7] formalized the information in terms of complexity. Their approach has led to a new area in information theory , called the algorithmic approach to information theory. The study of this approach is important to us because it gives the relation between randomness and compressibility.

1.2 Modelling of a stochastic time series:

In this section section two examples are presented which show that by using a deterministic model to represent a random series compression cannot be achieved.

1.2.1 Chaitin's Example:

In this section the relation between randomness and compressibility is described. The following discussion is taken from the works of Chaitin, Kolmogorov [5,6,7].

Almost everyone has an intuitive notion of what random number is. For example consider the following two series of binary digits.

0101010101010101010101

0110011001100001101 0

The first one is obviously constructed according to a simple rule. Inspection of the second series of digits yields no comprehensive pattern. The arrangement seems haphazard. In otherwords it is a random assortment of ones and zeros. Consider a probabilistic experiment of tossing a fair coin 20 times. The two sequences given above occur with a probability of 2^{-20}. If the above sequence is an outcome of a random experiment, then both these sequences, would have to be considered as random and indeed so, all the other sequences. But this clearly goes against our intuition.

Chaitin tried to define randomness in terms of the pattern it

follows. Thus a sequence is said to be random if the length of the shortest description for the sequence is equal to the length of the sequence itself, *i.e.* , for a random sequence it is better to describe the entire sequence as it is, rather than searching for a pattern. Thus randomness and compressibility are inversly related.

Consider a machine M, which takes sequences over a finite alphabet as input and outputs a program which contains a description of the sequence. The shorter the program is, the higher the compressibility of the sequence and *vice versa*. When a random sequence is given as input to this machine, the output program will have a length equal to the length of the sequence.

Definition 1.1: A sequence is said to have a complexity j if the length of the shortest program required to describe the sequence is equal to j.

Theorem 1.1: Consider a machine which takes sequences of length n, over a finite alphabet of cardinality p , and outputs a sequence, (whose values were assumed to be taken from the same alphabet as the input sequence)which represents the shortest description of the input sequence. The number of sequences which are of complexity less than j are $\frac{p^{j}-1}{p-1}$

Proof: The number of sequences having a complexity less than j are

$$1+p+p^{2}+\ldots+p^{j-1} = \frac{p^{j}-1}{p-1} \qquad \ldots(1.1)$$

One can clearly notice, that the fraction of sequences having a

complexity less than j is very small for very large lengths of the sequence. This implies that for large values of n higher compression ratios cannot be achieved unless we have a probabilistic description of the given data.

1.2.2 Gaines' model:

Consider a second example. Suppose that we are observing a time series of 0's and 1's as the behaviour of certain dynamical system- the complexity of which we intend to compress. Suppose the modeler assumes a priori that the observed series is generated by a causal system and therefore forms a deterministic model of it. For any finite length N of the observed time series, there will be an unbounded number of finite state machines whose input-output behaviour, starting from a specific state, is identical to that observed; out of these models there will be some such that no other machine has a smaller number of states. An optimal causal model is that which is a minimal-state machine model.

Consider binary sequences of length N. There are 2^N such sequences. Let us enumerate the different S-state deterministic automata that act as models for these sequences. The general form of the model will be a transient chain followed by a cycle. If there are S states to be filled with two symbols 0 and 1, and the cycle can commence in any state, there are atmost $S2^S$ models. Some of these models may generate the same sequence, making an upper bound on the number of models, so

$$M = S2^S \geq 2^N \qquad \ldots(1.2)$$

Now we will evaluate the mean number of states in the minimal form of these M models.

Let M(R) be the probability of finding models with states from R=1 to R. The average value of M(R) for R between 1 and S-1 is given by,

$$\bar{M}(R) = \frac{1}{S-1} \sum_{R=1}^{S-1} \frac{\Delta M(R)}{\Delta(R)} R \qquad \ldots(1.3)$$

where $\frac{\Delta M(R)}{\Delta(R)}$ is the fraction of models with state R

$$\text{Therefore } \bar{M}(R) = \frac{1}{s-1} \sum_{R=1}^{S-1} R[R.2^R - (R-1).2^{R-1}] \qquad \ldots(1.4)$$

Let μ_s be the average number of the state S. Since there are $\mu_s 2^{\mu_s}$ models like that,

$$\mu_s 2^S > \mu_s 2^{\mu_s} > \bar{M}(R) \qquad \ldots(1.6)$$

$$\text{or } \mu_s > \frac{1}{(S-1).2^S} \frac{1}{2} \sum_{R=1}^{S-1} (R^2 2^R + R2^R) \qquad \ldots(1.7)$$

Gaines' simplification of the summation in the above inequality was wrong. His simplification gave the following inequality,

$$\mu_s > S-2 + \frac{2}{s-1} - \frac{2}{(s-1).2^{s-1}} \qquad \ldots(1.8)$$

but we found that

$$\mu_s > \frac{1}{2}\left(S-2+\frac{2}{s-1}-\frac{2}{(s-1)2^{s-1}}\right) \qquad \ldots(1.8)$$

from the inequality (1.2)

$$S > N - \log_2 N - 2 \qquad \ldots(1.9)$$

Therefore,

$$\mu_s > \frac{s-2}{2} > \frac{N-\log_2 N-2}{2} \qquad \ldots(1.10)$$

$$R_N = \frac{\text{Expected number of states of models}}{\text{Number of observations}} = \frac{\mu_s}{N}$$

$$> \frac{1}{2} - \frac{\log_2 N+2}{N} \qquad \ldots(1.11)$$

Since Gaines ignored $\frac{1}{2}$ in the summation, he concludes that $R_N \to 1$ as $N \to \infty$. But the result we have got says that we can get compressions upto 50%, by using this model. This is due to the fact that the RHS in the inequality is a lower bound and is heavily underestimated.

The conclusion derived from the above two examples is that for large values of N it is difficult to form a causal model for the given series whose complexity is less than N. Infact we can say that, such a model doesn't exist, because the existence of such a

model clearly violates the fundamental theorem of information theory, which states that the entropy is a lower bound for the expected complexity of a code.

To alleviate this problem, we have taken the following steps.

(i) The sequence is broken into subsequences of smaller length. Each block of these sequences is separately coded.

(ii) An error criterion was selected, so that the reconstructed sequence at the decoding end will be within some error limit of the original sequence. This gives an improvement in the compression ratios at the expense of losing some information which for the objective at hand can be tolerated by a machine or a human. This technique can be mainly used in the coding of image and speech signals, in the absence of the knowledge of the statistics of the given signal.

We have considered partial realizations as an attractive choice for the aforesaid problem. Though they may not be optimal in Chaitin's sense of shortest description, the algorithms are simple and hence the choice.

1.3 Statement of the problem:

The problem that is attempted in this thesis is to study the feasibility of some well known representations from linear system theory *viz.* rational approximations to data with a fidelity criterion. Our aim is to fix the amount of distortion and to try to achieve higher compression ratios. Similar techniques have been employed in the probabilistic source coding techniques by placing an upper bound on the amount of distortion and minimising the

entropy [2]. Since our techniques are data dependent it is difficult to rank these with the existing techniques.

1.4 Overview of the thesis:

Chapter 2 describes the 1-D partial realizations, Berlekamp-Massey algorithm, the expected complexity of the rational approximation scheme, the introduction of error criterion for achieving data compression.

Chapter 3 describes an introduction to 2-D systems, 2-D Partial realizations , Sakata's extension of Berlekamp-Massey algorithm to two dimensions .

Chapter 4 describes the applications of 1-D and 2-D partial realizations to data compression, the drawbacks of these representations and possible improvements.

The conclusions are presented in chapter 5.

CHAPTER 2

1-D PARTIAL REALIZATIONS WITH AN ERROR CRITERION

2.1 Introduction:

The objective of the present chapter is the study of the models which view the given data as the impulse response of a minimum order discrete linear filter with data and the coefficients confined to a finite field. This representation is referred to as **minimal partial realization.**

The minimal partial realization is data specific and provides a transfer function P(Z)/Q(Z) of a linear discrete filter with degree of P(Z) less than that of Q(Z) and Q(Z) having a degree which is minimal over the family of all such realizations. The problem may not admit of a unique solution and many such realizations may be possible. Since with each rational function P(Z)/Q(Z) we may associate a formal power series by means of a long devision, e.g,

$$\frac{P(Z)}{Q(Z)} = a_1 Z^{-1} + a_2 Z^{-2} + \ldots + a_N Z^{-N} \qquad (2.1)$$

{$a_1, a_2, \ldots\ldots, a_N$ represent the impulse response sequence}

This implies that for a given sequence $a_1, a_2, \ldots\ldots, a_N$ P(Z)/Q(Z) is a realization if and only if the first N coefficents of the formal power series of P(Z)/Q(Z) are exactly the numbers $a_1, a_2, \ldots\ldots, a_N$. In this sense $a_1, a_2, \ldots\ldots, a_N$ is viewed as a *partial sequence* or a subsequence of a certain fixed sequence and

P(Z)/Q(Z) as a partial relization.In this sense the partial realization problem is the same as Padé representation problem. To obtain the minimal partial realization, we must some how find the right infinite extension sequence among many that may be possible. In this process we must not make any extrinsic assumptions. The problem of partial realization has been tackled by a number of alternate methods. For a definite treatment of these methods from the view point of abstract algebraic system theory, the interested person may refer to [14].Of great relevence to the present discussion , is due to Kalman[15,16], which attempts to provide an algebraic structure to all formal power series, by describing them in terms of **jump points** which constitute its **invariants**. Kalman shows that

(i)Every formal power series has a representation as an infinite continued fraction.

(ii)A formal power series is rational if and only if its continued fraction representation is a terminating fraction.

(iii) All the invariants of the power series can be generated by Euclidean algorithm.

Interestingly the same problem has been tackled in coding theory literature by **Berlekamp-Massey** algorithm[3,4]. The Berlekamp-Massey algorithm is a recursive design procedure for the design of a shortest linear feedback shift register [LFSR] that generates a given sequence. One can find a complete parellel of notion of jump and jump points in terms of L(n) *vs* n, where L(n) is the length of shortest LFSR that generates the sequence

$a_1, a_2, \ldots\ldots, a_n$ the partial sequence of the specified sequence $a_1, a_2, \ldots\ldots, a_N$; $n < N$. In this connection we state some results.

Let $a_1, a_2, \ldots\ldots, a_N$ be a specified sequence of length N. The problem is to design an LFSR of minimal length L as shown in Fig.2.1 such that the first N outputs of LFSR starting from the initial state $(a_1, a_2, \ldots\ldots, a_l)$ are exactly the specified sequence. An LFSR is completely charecterized by its length and the connection polynomial

$$\sigma(x) = 1 + \sigma_1 x + \ldots\ldots + \sigma_l x^l \qquad \ldots\ldots (2.2)$$

Let us denote it by $(L, \sigma(x))$. Let $(L(n), \sigma^{(n)}(x)$ be the shortest length LFSR that generates the partial sequence $a_1, a_2, \ldots\ldots, a_n$. The unknown parameters $L(n)=l, \sigma_i^{(n)}$ must satisfy the equation

$$\begin{bmatrix} a_1 & a_2 & \cdots & a_l & a_{l+1} \\ a_2 & a_3 & \cdots & & \\ \cdots & \cdots & \cdots & & \\ \cdots & \cdots & \cdots & & \\ a_{n-1} & \cdots & \cdots & a_n & \end{bmatrix} \begin{bmatrix} 1 \\ \sigma_1^{(n)} \\ \cdot \\ \cdot \\ \sigma_l^{(n)} \end{bmatrix} = 0 \qquad 2.3$$

If we denote the above coefficient matrix which has the Hankel form as $A(n-1, l)$, then we can prove the following results:

1. $L(n) = n/2$ if and only if $A(n/2, n/2)$ is nonsingular.
2. LFSR $(L(n), \sigma^{(n)}(x))$ is unique if $L(n) \leq n/2$ or stated in

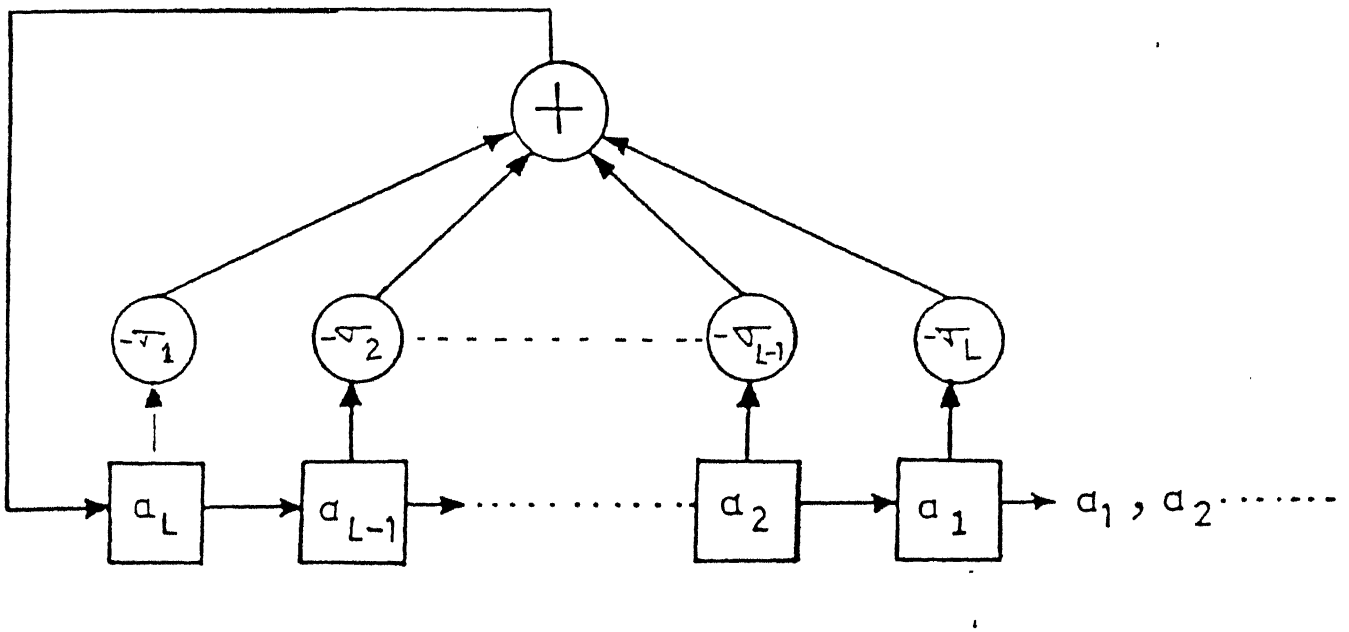


Fig 2.1. LFSR of length L that generates the given sequence.

another way if L(n)=m < n/2, then $(L(n), \sigma^{(n)}(x)) = (m, \sigma^{2m}(x))$

3. If n=2l' and n=2l(l>l') are two adjacent points where L(n) becomes n/2, then in the interval 2l' < n < 2l, L(n) jumps at only one point n = l+l' and the amount of jump is equal to l-l'.

4. The LFSR $(L(n), \sigma^{(n)}(x))$ has 2L(n)-n degrees of freedom in the sense that this many successive σ_i's may be chosen arbitrarily.

2.2 Berlekamp-MasseyAlgorithm:

Referring to equation (2.1)

if

$$P(Z) = p_o + p_1 Z + \ldots\ldots + p_m Z^m \qquad \ldots(2.4)$$
$$Q(Z) = q_o + q_1 Z + \ldots\ldots + q_n Z^n$$

We normalize the coefficients such that $q_o=1$. Now by formally equating the terms in eq (2.1) we get

$$\sum_{i=o}^{i=n} q_i a_{l-i} = 0 \quad \text{for } l=n+1, n+2, \ldots N \qquad \ldots(2.5)$$

$$\sum_{i=o}^{i=l-1} q_i a_{l-i} = p_l \quad \text{for } l=1, 2, \ldots\ldots m \qquad \ldots(2.6)$$

The coefficients of Q(Z) are obtained using Berlekamp-Massey algorithm as explained below.

The equation (2.5) represents an autoregressive filter, which can be implemented using shift register configuration as shown in Fig 2.1. To design this shift register we have to determine two

quantities.

(i) The shift register length n, which is the degree of the feedback polynomial Q(Z) such that n should be minimum.

(ii) The feed back coefficients $q_0, q_1, \ldots q_m$.

The design procedure is recursive. For each r, starting with the iteration r=1, we will design a shift register for generating the sequence , $a_1, a_2, \ldots . a_r$. Shift register $(n_r, Q^{(r)}(Z))$ is a minimum length shift register. This configuaration need not be unique for a given length n_r, and there may exist several such shift registers, all of the same length. At the start of the iteration r, we would have constructed a list of shift registers

$(n_1, Q^{(1)}(Z))$

$(n_2, Q^{(2)}(Z))$

.

.

$(n_r, Q^{(r)}(Z))$

The Berlekamp-Massey algorithm computes a new shortest length $(n_r, Q^{(r)}(Z))$, that generates the sequence $a_1, \ldots, a_r$ with the initial conditions $a_1, \ldots a_{n_r}$. This is done by using the most recent shift register, modifying the length if necessary.

At iteration r, compute the next output of the (r-1)th shift register.

$$\hat{a} = -\sum_{j=1}^{j=l} q_j \, a_{r-j} \qquad \ldots(2.7)$$

Let Δ_r be the difference between the desired output a_r, and the

actual output $\hat{a}_r$ of the shift register.

$$\Delta_r = a_r - \hat{a}_r = a_r + \sum_{j=1}^{j=l} q_j^{(r-1)} a_{r-j} \qquad \ldots\ldots(2.8)$$

equivalently

$$\Delta_r = \sum_{j=0}^{j=l} q_j \, a_{r-j} \qquad \ldots\ldots(2.9)$$

If $\Delta_r = 0$ then set $(n_r, Q^{(r)}(D)) = (n_{r-1}, Q^{(r-1)}(D))$ and the iteration is complete.

If $\Delta_r \neq 0$ then n_r and $Q^{(r)}(D)$ are modified as follows.

$$n_r = \max[n_{r-1}, r - n_{r-1}] \qquad \ldots\ldots(2.10)$$

$$Q^{(r)}(D) = Q^{(r-1)}(D) + AD^l Q^{(m-1)}(D) \qquad \ldots\ldots(2.11)$$

where,

$A = -\Delta_r / \Delta_m$.

$l = r - m$.

m = maximum m_i ; such that $m_i < r$ and $m \neq 0$.

One can easily prove that the new shift register $(n_r, Q^{(r)}(D))$ indeed generates the sequence $a_1, \ldots\ldots, a_r$ with the initial conditions $a_1, \ldots\ldots, a_l$. For proof, interested persons may refer to [4].

The algorithm is described with the help of a flow chart as shown in Fig 2.2.

From equation (2.10) one can clearly observe the jump behaviour of n . The variation of the linear complexity by

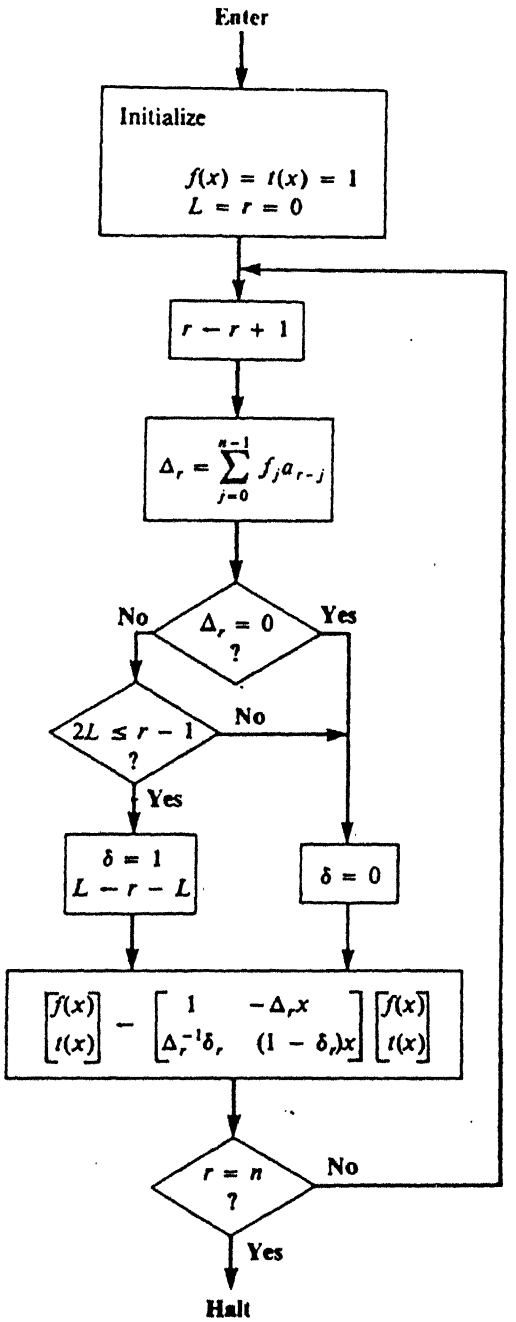


Fig 2.2. Berlekamp-Massey algorithm
(Reproduced from [4])

advancing each iteration is called linear complexity profile and is illustrated in Fig.2.3.

After obtaining the coefficients of Q(Z) we substitute that in equation(2.6) to get the coefficients of P(Z). Thus the partial realization is obtained.

The Berlekamp-Massey procedure will have multiplications of the order $O(n^2)$.

2.3 Introduction of Error criterion:

From the discussion in chapter 1, we notice that most of the sequences will have higher complexity. Therefore, we expect not to achieve high compression ratios, using partial realizations, which give the exact sequence upon reconstruction. In this section we prove that that is indeed true ,and suggest an alternate method.

2.3.1 Expected complexity of the partial realization procedure:

Before evaluating the expected complexity of the partial realization scheme , let us first evaluate the distribution of the lengths of the minimal LFSR 's. The following discussion is from [10]

Lemma 1:Consider sequences of length n which take values from a finite field GF(p). The distribution D(n;1) of the minimal LFSR 's is given by

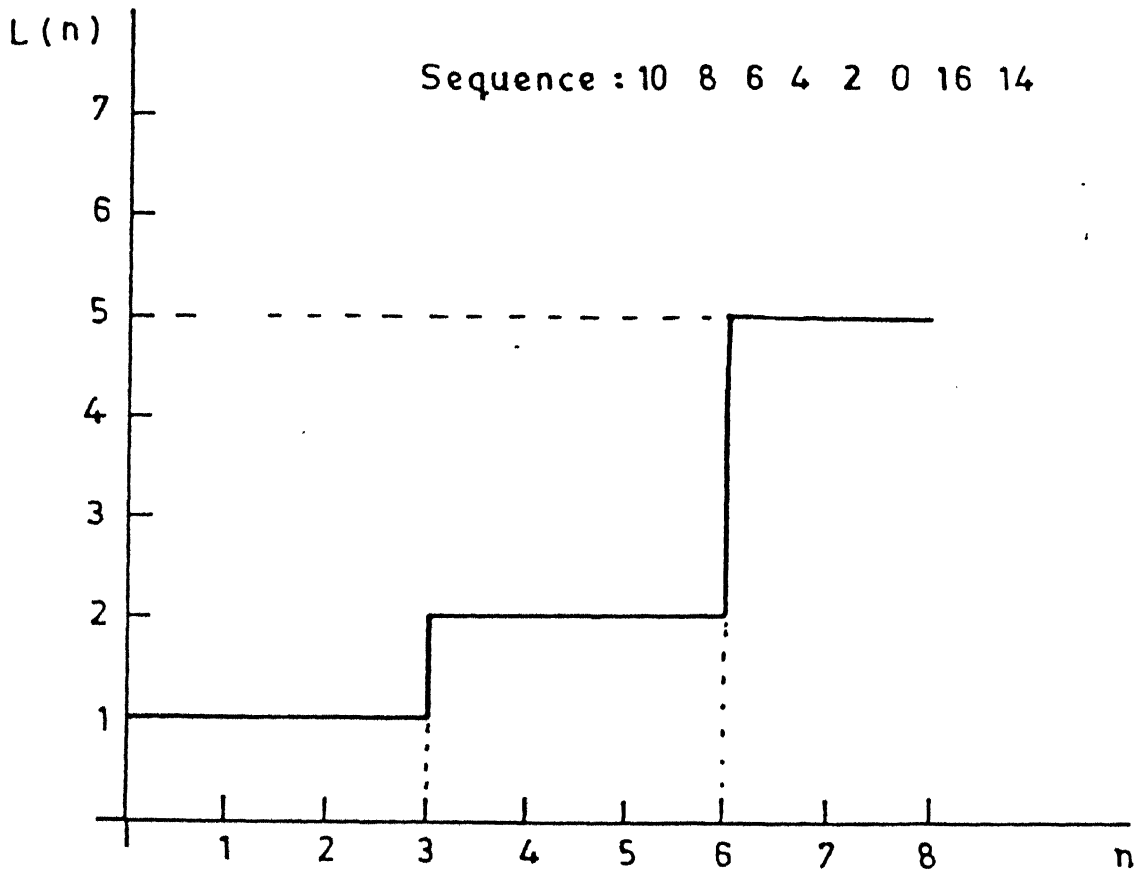


Fig 2.3. Linear complexity profile of a sequence

$$D(n;l) = \begin{cases} 1 & \text{if } l=0 \\ p^{2l-1}\bar{p} & \text{if } l=1,\ldots\ldots,\alpha \\ p^{2(n-l)}\bar{p} & \text{if } l=\alpha+1,\ldots\ldots,n \end{cases} \qquad \ldots(2.12)$$

where, $\bar{p} = p-1$

$\alpha = n/2$ if n is even

$= (n-1)/2$ if n is odd

Proof: The proof is by induction.

Equation (2.12) is true for n=1. Suppose (2.12) is also true for n=k and consider vectors $a_1, a_2, \ldots\ldots, a_k, a_{k+1}$. Here a_{k+1} can take p values and $\bar{p}$ of these values cause the feed back equation to fail. For these cases l must be corrected. On the other hand a_{k+1} satisfies the feedback equation for exactly one value and then $L_{k+1}(a) = L_k(a)$. It follows that p^k vectors retain the same length and the same set of initial conditions. The remaining $p^k\bar{p}$ vectors get a new set of initial conditions if there is a possible length change. Consider 3 cases.

case1: $0 \leq l \leq \alpha$

By induction there are $p^{2l-1}\bar{p}$ vectors whose minimum length generators have length l. Letting a_{k+1} arbitrary we have $p^{2l}\bar{p}$ vectors $a_1, a_2, \ldots\ldots, a_k, a_{k+1}$ to consider. Of these $p^{2l-1}\bar{p}$ remain of length l; $p^{2l-1}\bar{p}^2$experience a length change to k+1-l.

case2: $k+1-\alpha \leq l \leq k+1$

By induction there are $p^{2(k-l)}$ vectors $a_1, a_2, \ldots\ldots, a_k$ of formula length l; there is no vector of of formula length l=k+1. Adding component a_{k+1} , we have $p^{2(k-l)+1}\bar{p}$ vectors to consider. None of these experience a length change because l> k+1-l. Summing

these two components we obtain $p^{2(k+1-l)}\ \bar{p}$ formulas of length l.

case3: $\alpha < k-\alpha = l$

This case occurs only when k is odd. We consider formulas of length $l=\alpha+1$, the term missing in case2. By induction there are $p^{2(k-\alpha-1)}$ of this length. Adding component a_{k+1}, we obtain $p^{2(k-l)}\bar{p}$ formulas to consider. All have length $l=\alpha+1$ because $2(\alpha+1)=k+1$.

Combining cases (1),(2),(3) we see that equation (2.12) holds with n replaced by k+1. By induction it holds for any n. □

Now to compute the expected value of the length of the partial realization scheme, let assume that the sequences are equiprobable. Then the expected complexity E(n) is

$$E(n) = \left(\sum_{l=0}^{l=n} 2l\ D(n;l) \right) / p^n \qquad \ldots(2.13)$$

The factor 2 is due to the fact that we have to also consider the numerator coefficients.

Substituting the value D(n;l) from (2.12),(2.13) and after some algebraic manipulation we get

$$E(n) = n+(N/D) \qquad \ldots(2.14)$$

where $N < 2p^{n+1}$ if n even

$< p^n(1+p^2)$ if n odd

$N \geq 0$

and,

$D=(p+1)^2p^n$

We see that by using the above representation we are actually

expanding the data. However, since some error can be tolerated in the reconstructed sequence, we try to find a partial realization which, when expanded as a formal power series, gives a sequence, which is within some error limits of the original sequence. Thus we resort to a modified Berlekamp-Massey algorithm by introducing an error as described in [1,18].

The Berlekamp-Massey algorithm gives the transfer function realization of a filter, which when excited by an impulse, regenerates the data exactly. It must be noted that at the (r+1)th stage, if $Q^{(r)}(Z)$ was the denominator polynomial (Eq 2.1) to calculate $\hat{a}_{r+1}$ the predicted (r+1)th value of the data, and it turns out to be equal to a_{r+1}, i.e. if the error Δ_{r+1} =0, then

$Q^{(r+1)}(Z)$ =$Q^{(r)}(Z)$ and we forego the usual multiplication process, for the (r+2)th stage. This occurs rarely in a practical situation. We, however, allow for an error in the predicted values, such that,

$$a_{r+1} - \hat{a}_{r+1} \leq \varepsilon \qquad \ldots(2.15)$$

where ε is the maximum allowable error.

We now substitute $a_{r+1} = \hat{a}_{r+1}$, Δ_r =0 and do not modify the coefficients. If Δ_r >= ε , we use the normal Berlekamp-Massey procedure to obtain Q(Z). The entire procedure is described with the help of a flow chart as shown in Fig 2.4

The coefficients of P(Z) are then found using eq.2.6

Examples:

(i) Error =0

Modulo field =97

Desired sequence: 16,10,37,52,55,.....

Transfer function obtained using BM algorithm

$$H(Z) = \frac{1+84Z+9Z^2}{1+68Z+75Z^2+23Z^3}$$

(ii) Error =15 Modulo field =97

Desired sequence: 16,10,37,52,55,.....

Transfer function obtained using BM algorithm

$$H(Z) = \frac{1}{1+77Z}$$

Generated sequence: 20,12,46,47,67,....

Error sequence: 4,2,9,5,12,....

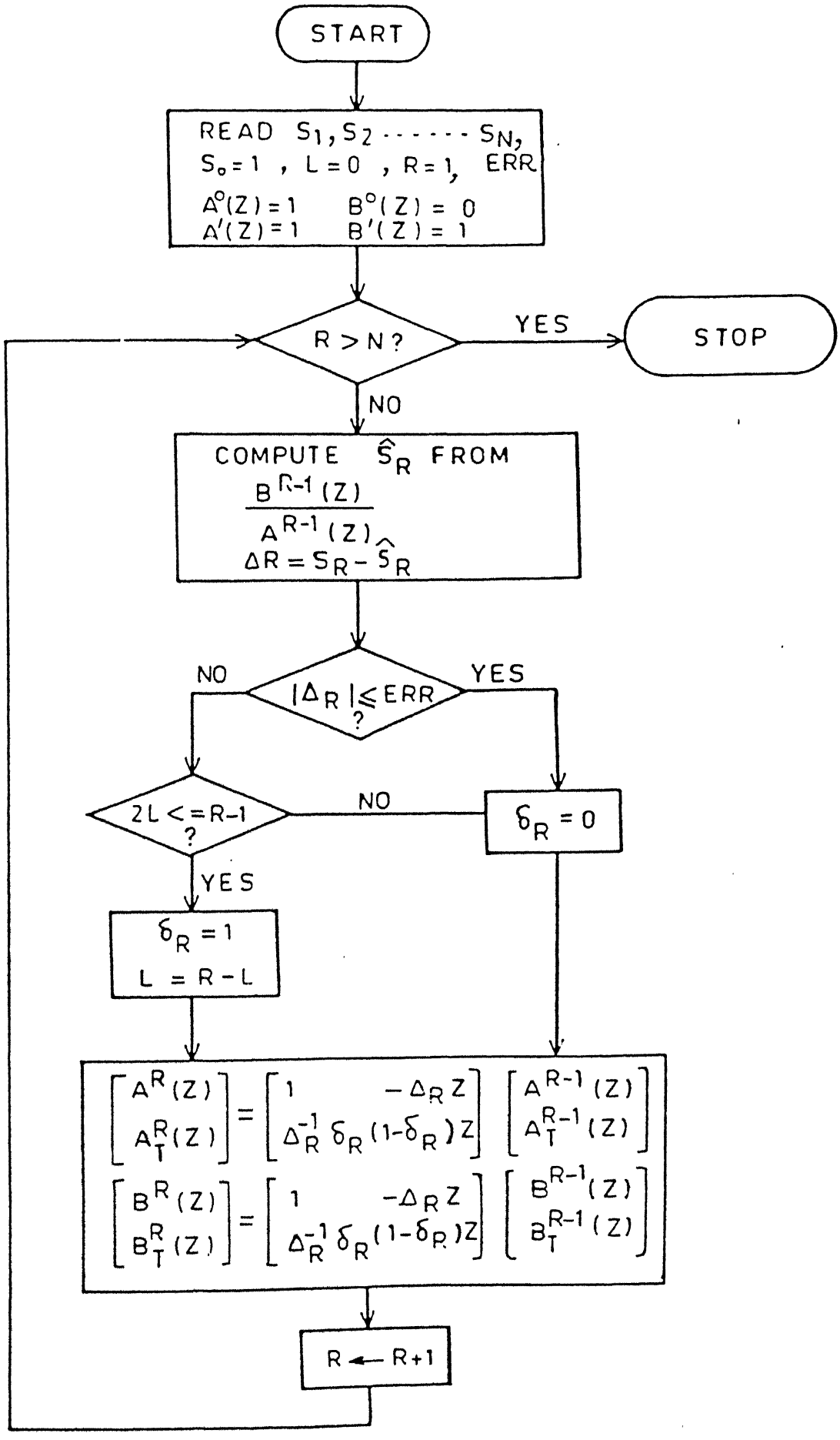


Fig 2.4. Modified Berlekamp-Massey Algorithm with error bound (Reproduced from [18])

CHAPTER 3
2-D PARTIAL REALIZATIONS

In this chapter we study the representation of signals as impulse response sequences over a finite field of a 2-D linear shift invariant system. We present the basic structure and properties of the two dimensional systems, and describe Sakata's extension of Berlekamp-Massey algorithm to two dimensions, for finding a partial realization to the given series.

3.1 Structure and properties of two dimensional systems:

In this section the basic structure and properties of two dimensional systems are briefly discussed. For a more detailed discussion interested person may refer to [8].

The first contribution that discussed the problem of defining dynamical systems, was motivated by the necessity of investigating recursive structures for the two dimensional data, such as images and seismic signals. The idea of input-output description of systems in two indeterminates as well as the design of analytic techniques based on the frequency response and two dimensional Z-transforms is well known. The new ideas that promoted further research in two dimensional systems for the concept of the state and its updating equations.

From the beginning, very deep and substantial differences from the theory of dynamical systems in one variable have been evident. These are due to the mathematical tools, the concept of state itself, and to the structure of the state updating

equations.

3.1.1 Two dimensional Input-Output maps:

Let $\mathcal{T}$ denote the cartesian product ZxZ, partially ordered by the product of orderings, and introduce the notion of past and future of a point $p := (p_1, p_2)$ in $\mathcal{T}$ as shown in Fig. 3.1. We call the past of p the set

$$\mathcal{P}_p = \{(i,j);\ p_1 \geq i,\ p_2 \geq j,\ (i,j) \neq (p_1,p_2)\} \qquad \ldots(3.1)$$

and the future of the point p, the set

$$\mathcal{F}_p = \{(i,j);\ p_1 \leq i,\ p_2 \leq j\} \qquad \ldots(3.2)$$

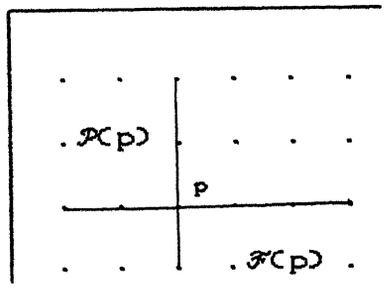


Fig 3.1 Past and future of a point which do not divide the two dimensional plane into seperation sets.

Definition 3.1: A two dimensional system in its Input output form is defined as the set $H=(\mathcal{T}, K, \mathcal{U}, \mathcal{Y}, F)$, where

$\mathcal{T}$ is the time set defined above.

$\mathcal{U}$ and $\mathcal{Y}$ are the input and output spaces over a field K

F is the input-output map.

F is assumed to satisfy the following axioms.

1. Linerity

2. Two dimensional shift invariance.

But such a concept of past and future in two dimensions is not sufficient for the description of the system because it doesn't divide the two dimensional plane into two seperate sets. Therefore it is necessary that we introduce another notion of past and future so that the two dimensional plane can be divided into separate sets. This is necessary because we should be able to reconstruct every point to the future of (m,n), where (m,n) is the dimension of the system, using the past values upto the point (m,n). Kung et. al.[17], gave a definition for the past and future of a point. According to their definition the past of a point $p=(p_1,p_2)$ is defined as

$$\mathcal{P}(p) := \{(i,j) \mid (i+j) \leq (p_1+p_2)\} \qquad \ldots(3.3)$$

and the future of the point $p=(p_1,p_2)$ as

$$\mathcal{F}(p) := \{(i,j) \mid (i+j) > (p_1,p_2)\} \qquad \ldots(3.4)$$

The seperation of the two dimensional plane using the above definition is illustrated in the Fig. 3.2.

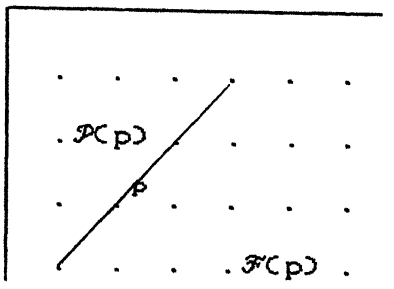


Fig. 3.2 Past and future of a point which divide the two dimensional plane into seperation sets

Kung et. al[17], have proved that every formal power series in two variables can be expressed as rational approximation if and only if the associated Hankel matrix

$$\begin{bmatrix} a_{00} & a_{01} & a_{10} & a_{20} & \cdot & \cdot & \cdot & \cdot \\ a_{01} & a_{10} & \cdot & \cdot & \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & & \\ \cdot & \cdot & \cdot & \cdot & \cdot & & & \end{bmatrix}$$

is of finite rank.

But even by this new concept of defining past and future, some of the problems will remain unresolved. One such problem is the inability to check the recurrence relation (a relation by which we will be able to find the value of the sequence at a point using the past values till that point) at some of the points even though they are the future points with respect to the dimension of the system. We illustrate this using an example.

Consider a series $(a)_{i,j}$ $i \geq 0, j \geq 0$; which can be reconstructed

using a linear recurrence relation represented by a polynomial in two variables $z_1^2 z_2^2+1$ which represents a linear recurrence of the form,

$$a_{i-2,j-2}+a_{i,j} = 0 \quad \text{for all } (i,j)\in\mathcal{F}((2,2)) \qquad \ldots(3.5)$$

Consider applying this relation at the point $(5,0)\in\mathcal{F}((2,2))$

The required relation is $a_{3,-2}+a_{5,0}=0$.

But $a_{3,-2}$ is not defined. In order to avoid this problem we have two options.

1. We can consider $a_{i,j}$ to be 0 for $i<0$ or $j<0$. However it is not feasible since the system dimensions tend to increase.

2. The approach we have followed is taken from the discussions of Sakata[20,21,22,23] in the context of application to coding theory. Sakata has illustrated that a set of polynomials with degrees of each polynomial minimal,(The actual definition of minimality comes later in this chapter) and a set of initial conditions based on the degrees of the polynomials will be sufficient to construct the remainig sequence.

In this chapter we will study the approach used by Sakata. The problem that is attempted here is to find a set of linear recurrences of the form,

$$\sum_{r\in\Gamma_f} f_r\, a_{r+q-s} = 0 \qquad \ldots(3.6)$$

where s is the degree of the polynomial, q is a point in the two dimensional plane $\geq$ s and Γ_f a subset of the two dimensional plane over which f_r is nonzero.

Note: In the above equation r,q, and s are not scalars, but

two-tuples. If $s := (s_1, s_2)$ and $t := (t_1, t_2)$, then s+t is defined as (s_1+t_1, s_2+t_2).

The ordering $\geq$ is defined as follows.

$s := (s_1, s_2) \geq t := (t_1, t_2)$ if and only if s_1 is not less than t_1 and s_2 is not less than t_2.

We have to find a set of minimal degree polynomial and a set of initial conditions which will determine the sequence uniquely.

3.2 Definitions and Notations:

Let K be any field. Over the 2-D lattice $\Sigma_o = Z_o^2$ a total order $\leq_T$ is defined as follows.

$0 := (0,0) <_T (1,0) <_T (0,1) <_T (2,0) \ldots\ldots$

and $p \leq q$ if and only if $p <_T q$ or $p=q$

One can number the points of the 2-D lattice Σ_o with respect to the total order defined above. For example $n_o((0,0)) = 0$, $n_o((1,0)) = 1$, $n_o((0,1)) = 2$ as shown in Fig. 3.3.

i \ j	0	1	2	3
0	.	.	.	.
1	.	.	.	
2	.	.		
3	.			

0	2	5	9
1	4	8	
3	7		
6			

Fig. 3.3. Total ordering on Σ_o

For a point $p := (p_1, p_2) \in \Sigma_o$, the next point p+1 with

respect to the total order $\leq_T$ is defined by

$$p+1 := (p_1-1, p_2+1) \quad \text{if } p_1 \geq 1$$
$$:= (p_2+1, 0) \quad \text{if } p_1 = 0 \qquad \ldots(3.7)$$

For $s,p,c \in \Sigma_o$ the following subsets of Σ_o are defined.

$$\Sigma_s := \{q \in \Sigma_o \mid s \leq q\}$$

$$\Sigma_s^p := \{ q \in \Sigma_s \mid q <_T p\} = \{q \in \Sigma_o \mid s \leq q <_T \}$$

$$\Gamma_c := \{ q \in \Sigma_o \mid q \leq c \} \qquad \ldots(3.8)$$

See Fig 3.4.

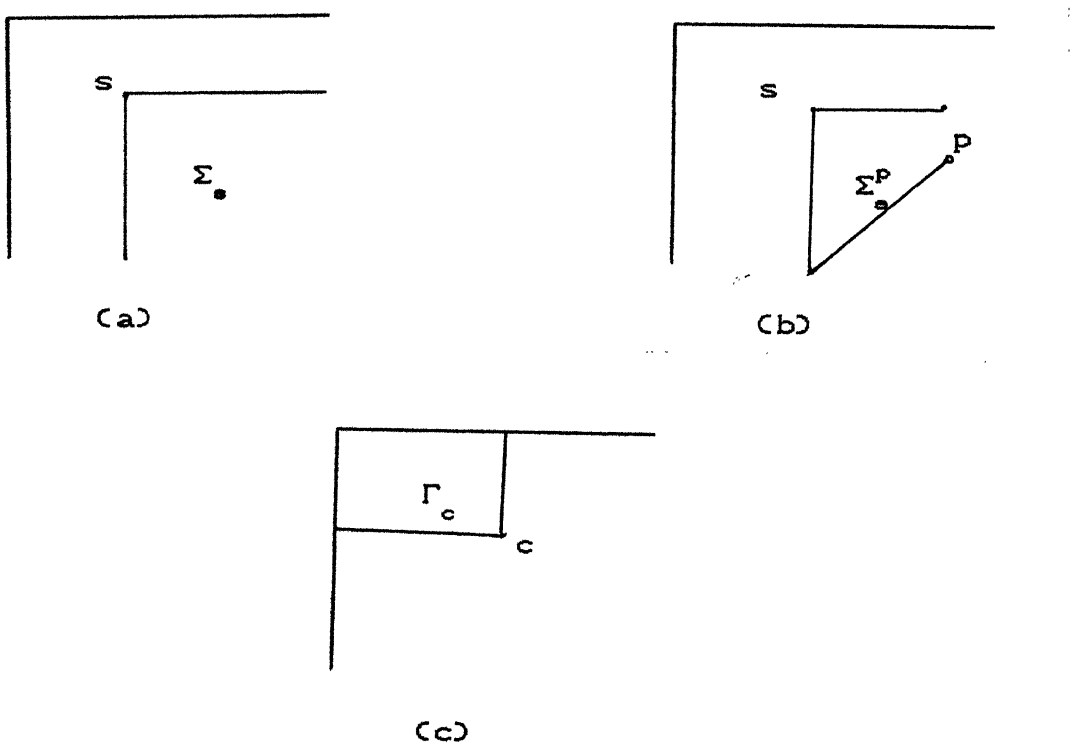


Fig. 3.4. Subsets of Σ_o (a) Σ_s (b) Σ_s^p (c) Γ_c

For an array $A := (A)_q$ (an array a with the support Σ_o^q) we consider a linear recurrence relation

$$\sum_{r \in \Gamma_f} f_r \, a_{r+q-s} = 0 \qquad \ldots(3.9)$$

where $f_r \in K$ and Γ_f is a finite subset of Σ_o; for $p := (p_1, p_2)$, $q := (q_1, q_2) \in \Sigma_o$, $p+q := (p_1+q_1, p_2+q_2)$. With the above linear recurrence relation a bivariate polynomial may be associated

$$f := \sum_{q \in \Gamma_f} f_q \, Z^q \in K[Z] = K[z_1, z_2] \qquad \ldots(3.10)$$

where $Z^q = z_1^{q_1} z_2^{q_2}$ is a power product of the independent variables z_1 and z_2

and,

$K[Z] = K[z_1, z_2]$ = Ring of bivariate polynomials over the field K

The degree Deg(f) of f is defined to be the maximum element of Γ_f with respect to the total order $\leq_T$ provided that

$$\Gamma_f := \{ q \in \Sigma_o \mid f_q \in K \neq 0 \}$$

Definition 3.2: For a finite array A, f is said to be valid upto p if and only if Eqn (3.9) holds for any point $q \in \Sigma_s^p$

Example: For a binary array shown in Fig.3.5.,

1 0 1 0 1 0 1 0

0 1 1 0 0 1 1 0

1 0 1 0 1 0 1 0

0 0 0 1 1 1 1 0

1 0 0 0 0 0 1 0

0 1 1 0 0 1 1 0

1 0 0 0 0 0 1 0

0 0 0 0 0 0 0 1

Fig. 3.5. A 2-D binary array

the polynomial $f = z_1^2 z_2^2 + z_2^2 + z_1 z_2 + z_1^2 + z_1$

with s = Deg(f) = (2,2) is valid at any point a(i,j) ≥ s i.e,

$$f[A]_{i,j} = a_{i,j} + a_{i-2,j} + a_{i-1,j-1} + a_{i-1,j-1} + a_{i,j-2} + a_{i-1,j-2} \quad \ldots(3.11)$$

3.3 2-D Berlekamp-Massey algorithm:

In this section the 2-D Berlekamp-Massey algorithm is explained. Proofs of theorems mentioned in this section can be found in [22].

Theorem 3.1: For a finite 2-D array A, if there is a polynomial g with deg(g) = t, which is valid upto q, there is no polynomial f with deg(f) ≤ t valid at every point $r \in \Sigma_s^q$ and also at q.

Definition 3.3: A polynomial g of deg(g)=t is said to have span(g) :=q-t if and only if g is valid at every point $r \in \Sigma_t^q$ and not valid at point q.

Definition 3.4: Excluded point set of $A = (a)_p$ is defined by

$$\Delta_e(A) = \bigcup_{t,q \in \Sigma_e}{}_p \; \Delta_{q-t} \qquad \ldots(3.12)$$

$$\Delta_{q-t} := \Gamma_{q-t} = \{r \in \Sigma_o \mid r \leq q-t\} \text{ if } \exists \text{ a } g \text{ with } span(g) = q-t$$
$$:= \Phi \quad \text{otherwise} \qquad \ldots(3.13)$$

Let C be the set of maximal points $\Delta_e(A)$

$$\text{Let } \Gamma_c := \bigcup_{c \in C} \Gamma_c \qquad \ldots(3.14)$$

We can associate a subset G of K[Z] corresponding to the set C of all maximal points $c \in \Delta(A)_p$, i.e , every element c of C is span(g) for some $g \in G$. G is called the auxiliary polynomial set associated to the minimal polynomial set $F \in FF(A)_p$.

Definition 3.5: For an array A VALPOL(A) is defined to be the set of all valid polynomials for the array A.

VALPOL(A) is an ideal of polynomial ring K[Z]. Thus if $f \in$VALPOL(A), then $gf \in$VALPOL(A) for any $g \in$K[Z].

Our aim is to find a set of minimal polynomials if and only if all the following conditions are satisfied.

1. $F \subset$ VALPOL(A)

2. $S := \{deg(f) \mid f \in F\}$ is **nondegenerate**, where a finite subset S of Σ_0 is said to be nondegenerate if and only if there are no distinct s and $t \in S$ and $s \leq t$. Therefore the set S will assume the form of a staircase as shown in Fig 3.6. The region $\Sigma_0 - \Sigma_s$ is denoted by $\Delta(F)$ also shown in Fig. 3.6.

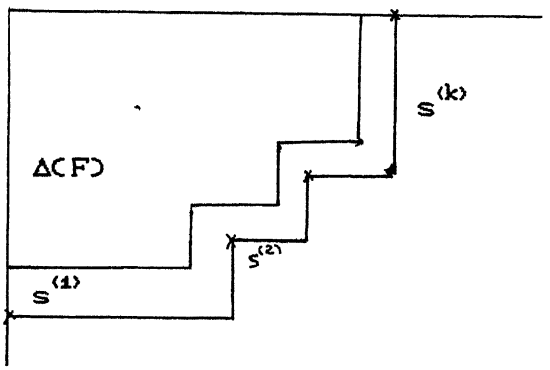


Fig. 3.6. The sets S and $\Delta(F)$ of a 2-D array $(A)_p$

Remark: The set $\Delta(F)$ and hence S is unique for an array A, but the set F is not unique. Hence from here we denote $\Delta(F)$ by $\Delta(A)_p$.

3.3.1 Description of the algorithm:

The 2-D Berlekamp-Massey algorithm is an iterative procedure for finding a minimal polynomial set $F \in FF(A)_p$ and the set $\Delta(A)_p$. Before going into the details the following theorem is stated without proof. Proof can be found in [22].

Theorem 3.2: $\Delta_e(A)_p = \Delta(F)$.

This implies that at point p we have an excluded point set, the maximal points of which are adjacent to the set S, the set of the degrees of valid polynomials at p.

The importance of the theorem can be explained as follows. We check whether polynomials which were found valid till the point p-1 are valid at point p. If some of them turn out to be not valid the new auxiliary polynomial set and hence the excluded point set $\Delta_e(A)_p$ can be found. By the adjacency relation which is a consequence of the theorem 3.2, the new degree of the polynomial can be found.

Theorem 3.3: Let $p <_T q$ and $g,f \in K[Z]$ with $\deg(g)=t$ and $\deg(f)=s$. If $g \in \text{VALPOL}(A)_p$, but $g[a]_p \neq 0$ and $f \in \text{VALPOL}(A)_q$, but $f[a]_q \neq 0$ i.e.,

$$\sum_{m \in \Gamma_g} g_m a_{m+n-t} = 0 \quad \text{if } n \in \Sigma_t^p$$
$$= d_p \neq 0 \text{ if } n=p \qquad \ldots(3.15)$$

$$\sum_{m \in \Gamma_f} f_m a_{m+n-s} = 0 \quad \text{if } n \in \Sigma_s^q$$
$$= d_q \neq 0 \text{ if } n=q \qquad \ldots(3.16)$$

then,

$$h := Z^{r-s} f - (d_q/d_p) Z^{r-q+p-t} g \in \text{VALPOL}(A)_{q+1} \qquad \ldots(3.17)$$

where,

$r := (r_1, r_2)$ is defined by

$$r_1 := \max(s_1, t_1+q_1-p_1),\ r_2 := \max(s_2, t_2+q_2-p_2) \qquad \ldots(3.18)$$

Remark: $t_1+q_1-p_1$, $t_2+q_2-p_2$ can be negative.

The polynomial h defined by the above equation may be written as $h = h(f,q,s,g,p,t)$.

An immediate consequence that follows from the above theorem is the following result.

Theorem 3.4: Let $s^{(i)} \in S$ be the elements of the set of degrees of $f^{(i)}$, where $f^{(i)} \in F$ are the minimal polynomials for the array $(A)_p$

If $p \in s^{(i)} + \Delta(A)_p$, then there exists a polynomial h such that

$h \in VALPOL(A)_{p+1}$ and $deg(h) = s^{(i)}$.

Based on the above theorems we will explain the actual algorithm and its complexity.

Define the following sets.

$F = \langle f^{(k)} \mid 1 \le k \le l \rangle \subset VALPOL(A)_q$

$S = \langle s^{(k)} = deg(f^{(k)}) \mid 1 \le k \le l \rangle$

G = Auxiliary polynomial set = $\langle g^{(k)} \mid 1 \le k \le l-1 \rangle$

$T = \langle t^{(k)} = deg(g^{(k)}) \mid 1 \le k \le l-1 \rangle$

PG = the set of points till which the polynomials $g^{(k)}$ are valid = $\langle p^{(k)} \mid 1 \le k \le l-1 \rangle$

DG = the errors at points $p^{(k)}$ by using the polynomials $g^{(k)}$

$= \langle d^{(k)} \mid 1 \le k \le l-1 \rangle$

The following provedure is an iterative procedure which gives the set of minimal polynomials valid till the desired point q.

Step1: n := (0,0); F=⟨1⟩; G=ϕ ; DG = ϕ ; PG = ϕ; S=⟨(0,0)⟩

T=ϕ

Step2: begin

step(i):find the polynomials which are not valid at the point n;

step(ii):find the polynomials whose degrees are such that if $n \le s^{(i)} + \Delta(A)_n$, replace them by new h 's

step(iii):For the remaining polynomials which satisfy step(i), but not step(ii); put them in a set F_{NN}, the point n in PG_{NN}, the errors in DG_{NN}, the degrees in S_{NN}.

Update the set G,PG,DG,T by

$G = G \cup F_{NN}$

$PG = PG \cup PG_{NN}$

$DG = DG \cup DG_{NN}$

$T = T \cup S_{NN}$

step(v): Find out the new $\Delta(A)_{n+1}$ from new P,G,and T.

step(vi): From the Δ set find the degrees of new minimal polynomial set by using the adjacency relation. $(\Delta_{\bullet}(A)_{n+1} = \Delta(A)_{n+1} = \Sigma_o - \Sigma_{\bullet})$

step(vii): from these new degrees update the polynomial set F by modifying the degrees and coefficients of polynomials in F_{NN} by using the Berlekamp-Massey procedure described in theorem.

step(viii): Now all the updated polynomials are put in the set F.

(Thus all the sets of importance are updated.)

Step3: n := n+1; if n=q, then stop; else go to step2:

Sakata has proved that this algorithm has a computational complexity of the order $O(n^2)$.

Now the sets F , S(or $\Delta(F)$) , with the initial conditions A with the support $\Delta(F)$ uniquely determine the sequence A upto the point q. A sample computation of the algorithm for the sequence shown in Fig 3.7, is shown in table3.1.

1 0 0 0 . . . .

0 0 0 . . . .

1 1 . . . .

1 . . . .

. . . .

Fig. 3.7. A binary 2-D array

Table 3.1 Sample computation of 2-D BM algorithm for the array shown in Fig. 3.7

n	F	G	P
(0,0)	$\{Z_1, Z_2\}$	{1}	{(0,0)}
(0,1)	$\{Z_1, Z_2\}$	{1}	{(0,0)}
(1,0)	$\{Z_1, Z_2\}$	{1}	{(0,0)}
(2,0)	$\{Z_1^2+1, Z_2\}$	$\{Z_1, 1\}$	{(2,0),(0,0)}
(1,1)	$\{Z_1^2+1, Z_2\}$	$\{Z_1, 1\}$	{(2,0),(0,0)}
(0,2)	$\{Z_1^2+1, Z_2\}$	$\{Z_1, 1\}$	{(2,0),(0,0)}
(3,0)	$\{Z_1^2+Z_1+1, Z_2\}$	$\{Z_1, 1\}$	{(2,0),(0,0)}
(2,1)	$\{Z_1^3+Z_1^2+Z_1, Z_1Z_2+Z_1, Z_2^2\}$	$\{Z_1^2+Z_1+1, Z_2\}$	{(2,1),(2,1)}

CHAPTER 4

APPLICATIONS OF PARTIAL REALIZATIONS TO DATA COMPRESSION

The objective of the present chapter is to study whether partial realization schemes can effectively be used to compress the data. We have applied this coding scheme on the image data.

4.1 Applications of 1-D partial realizations to image compression:

The coding scheme used here is the representation of signals over a finite field, which represent some realistic data such as, images etc., using 1-D partial realizations with error criterion. The sequence is represented by the coefficients of the polynomials P(Z),Q(Z). We have applied this coding scheme on image data. The image we have taken is a 64x64 image with 32 gray levels. The field in which the coefficients were found is GF(31). Coding is done individually on blocks of data of length 8. These blocks were scanned horizontally. The compression ratios obtained for various error values are shown in table 4.1

The reconstructed images are shown in figure 4.1. If we look at the table 4.1 we observe that the compression ratios obtained are very small. It is also possible that this sort of coding may lead to data expansion. Since the expected complexity analysis is very difficult with the introduction of the error criterion we have run some simulations on the computer. We have taken sequences of length 8 over a field GF(5) and evaluated the distribution of

Table 4.1 Compressions obtained for various error values using BM algorithm with error bound.

Error value	Compression %
2	6.6
4	8.8
5	11.7
7	11.9

the number of sequence for each length of the LFSR. We have chosen an error of 1. The results are shown in table 4.2.

Table 4.2 Distribution of LFSR lengths using BM algorithm with error bound

Length of the LFSR	Number of sequences
0	256
1	5576
2	13542
3	54072
4	188980
5	100513
6	22258
7	5244
8	384

Now if we compute the expected complexity E of the code,then

(a) Original image

(b) Compressed image
with error=2

(c) Compressed image
with error=4

(d) Compressed image
with error=5

(e) Compressed image
with error=7

Fig. 4.1 Compressed images using BM algorithm with error criterion

$$E = \left(\sum_{l=0}^{l=n} 2l D(n;l) \right) / p^n \qquad \ldots(4.1)$$

where D(n;l) is the number of sequences having a code length l, n is the length of the sequence, and p is the cardinality of the alphabet over which the sequences are defined.

Substituting in equation 4.1 the values of D(n;l) we get the value of E as E =8.3 ≥ 8. Thus we see that we cannot achieve compression even by introducing the concept of error criterion. Therefore, the following encoding scheme is chosen.

If the code length (number of coefficients of both numerator and denominator of the rational approximation, which is equal to twice the length of the LFSR) is greater than,the sequence is represented as it is. We encode only when the code length is less than the length of the sequence. Using this approach we got better results, as shown in table 4.3.

Table 4.3. Compressions obtained for various error values using modified representation of sequences using partial realizations with error criterion

Error value	Compression %
2	22
4	24
5	26
7	28

(a) Original image

(b) Compressed image
with error=2

(c) Compressed image
with error=4

(d) Compressed image
with error=5

(e) Compressed image
with error=7

Fig. 4.2 Compressed images using modified representation using partial realizations with error criterion.

4.2: Application of 2-D Partial realization to compression:

Application of 2-D partial realization schemes to compression have yielded very bad results. It was found that this scheme without introducing the error criterion will in general expand the data. Where as, this method is very useful for the decoding of multidimensional codes, they are ingeneral found to be very bad choice for data compression. This may be attributed to the use of a set of polynomials rather than a single polynomial. But as discussed in chapter 3 a single polynomial will not suffice to describe the entire sequence. Introduction of error criterion is also very difficult, because, while we aim to reduce the degree of one polynomial the degree of other polynomial may increase. Hence it can be concluded that 2-D partial realization is not a very good choice for partial realization.

4.3. Drawbacks of using partial realization schemes for data compression:

In the previous sections the application of partial realization schemes to data compression is described. From here onwards the discussion will be confined to 1-d case.

From the application of these algorithms to the image data the following observations are made.

(i) Higher compression ratios cannot be achieved even for high values of the errors.

(ii) The compresion ratio may not increase by increasing the value of the error.

In this section we prove that these representations are not optimal in the sense, that they are not minimum realizations such that the reconstructed signal is with in the error limits.

Notice that the linear complexity profile as described in chapter 2 exhibits a jump behaviour while using the Berlekamp-Massey algorithm. Suppose that at the r-th stage the number ofshift registers needed for the transfer function realization are l. Assume that at stage r, l to be less than (r+1)/2. At the (r+1)th stage if the predicted value of sequence $\hat{a}(r+1)$ is not equal to a(r+1) we find a new connection polynomial the degree of which jumps to (r+1-l) which is greater than l. This jump behaviour is illustrated in (fig)

We now proceed to prove that introduction of an error criterion as described in chapters 2 and 3 need not give an improvement Infact it is possible that it may even give worse results.

Suppose that we are finding the connection polynomial Q(Z) with an error ϵ. At the r-th stage let the number of shift registers needed be l_ϵ. Assume that

$$l_\epsilon^{(r)} < l_o^{(r)} < (r+1)/2 \qquad \ldots(4.2)$$

where,

$l_o^{(r)}$ = number of shift registers needed to reconstruct the exact sequence till r-th stage.

Let the predicted value $\hat{a}_{r+1}$ at the (r+1)-th stage differ

from a_{r+1} by more than ϵ , i.e ,

$$| a_{r+1} - \hat{a}_{r+1} | > \epsilon \qquad \ldots(4.3)$$

In this case the degree of the connection polynomial at the (r+1)-th stage with error is

$$l_{\epsilon}^{(r+1)} = r+1 - l_{\epsilon}^{(r)} \qquad \ldots(4.4)$$

and that without error is,

$$l_{o}^{(r+1)} = r+1 - l_{o}^{(r)} \qquad \ldots(4.5)$$

From equations (4.2),(4.4) and (4.5) it is clear that $l_{\epsilon}^{(r+1)} > l_{o}^{(r+1)}$. It means, that, for some sequences it may be possible that, the degree of connection polynomial with error criterion is greater than that, without the error. However, since such cases occur rarely, we expect on the average, that partial realizations which allow for some errors do achieve better compression than, the same methods which do not tolerate any error.This also explain why the compression ratios do not substantially increase with errors.

Since the Berlekamp-Massey algorithm does not give optimum results with an error criterion we search for an alternate formulation of the problem. Instead of updating the coefficients sequentially such that the predicted values are within the error bound, we pose the following question. Which one of the sequences which are with in the error bound has least linear complexity. That is, we want to find a transfer function realization of the form,

$$\frac{P(Z)}{Q(Z)} = \hat{a}_1 Z^{-1} + \hat{a}_2 Z^{-2} + \ldots + \hat{a}_N Z^{-N} \qquad \ldots(4.6)$$

such that,

$| a_i - \hat{a}_i | \leq \varepsilon$

and the degree of QCZ) minimum.

As the following theorem states the computation required to realize such a system must have an exponential complexity, which cannot be carried out by a nondeterministic polynomial time algorithm.

Theorem 4.1:Given a sequence $a_1, a_2, \ldots\ldots, a_n$ over a field GF(q) and an error limit ε such that $\varepsilon << q$ the problem of finding a minimal linear recurrence equation with least degreee such that the reconstructed sequence $\hat{a}_1, \hat{a}_2, \ldots\ldots, \hat{a}_n$ is within the error limiti.e.,

$| a_i - \hat{a}_i | \leq \varepsilon \ \forall \ i \in \{1,2,\ldots,n\}$

is NP-hard

Proof: In the worst case there are $(2\varepsilon+1)^n$ sequences which are within the error limit. In order to find the sequence that has got least linear complexity, we have to enumerate each of these sequences by Berlekamp-Massey algorithm (the B-M algorithm is optimal in number of computations to find the linear recurrence relation) and find the one with minimum linear complexity among these. Since size of the input itself is exponential in n, this problem is NP-hard. Therefore there is no other way to tackle this problem except by using a brute force method.

Using brute force approach we have coded the same image referred earlier. This approach gave a very high improvement over the previous two cases. Where as previously we could get a

compression ratio of 25% for an error 3 now we could get compression ratios upto 60%. The compression ratios obtained for various values of error are shown in table 4.4. The only problem with this approach is that for large values of error the computation is infeasible.

Table 4.4 Compressions obtained using brute force method.

Error value	Compression %
1	38
2	47
3	58

(a) Original image

(b) Compressed image
with error=1

(c) Compressed image
with error=2

(d) Compressed image
with error=3

Fig. 4.3 Compressed images using brute force method

CHAPTER 5

CONCLUSION

In the absence of knowledge ofthe statistics of the signal, it was generally observed that data compression is difficult to achieve. This fact was illustrated by using Chaitin's example. This difficulty arises because , the expected complexity of the sequence cannot be less than the entropy. However by using a fidelity criterion which for the objective at hand can be tolerated by a machine or human, it was conjectured that some compression will be achieved.

The signals that were considered in this thesis were sequences over a finite field GF(p) where p is a prime. The signals have been represented using partial realizations. The error criterion employed here was to place an error bound on the sequence, so that each value of the sequence estimated by this representation will be within the error bound of the original sequence. We have considered two cases.

1. One dimensional partial realizations.
2. Two dimensional partial realizations.

In first case the modified Berlekamp-Massey algorithm with error proposed by Reed was implemented. Application of this representation to image data showed that very little compression ratios would be obtained. By running some simulations it was shown

that, the data may actually expand using this representation even with an error criterion. A slight modification was suggested to this representation, that, when the length of the representations (number of coefficients needed for the representation) exceeds the length of the sequence, the sequence is represented as it is. It was illustrated that the representation using the error criterion may, sometimes give worse results than the representation without error, though, for very limited cases. An alternate formulation of the problem was chosen which seeks a representation with minimum number of coefficients such that the reconstructed sequence is within the error bound. It was proved that this problem is NP-hard and is computationally intractable, for large values of the sequence lengths and error bounds. However, the brute force method was tried on an image data for small values of the error. Compression ratios upto 60% were obtained, but with a very high computational cost.

2-D partial realizations using Sakata's extension of Berlekamp-Massey algorithm to two dimensions was implemented. The application of this representation to image compression yielded unpromising results. This may be attributed to over complex-view of the system under observation. Introduction of error bound is very difficult, because when one seeks to reduce the degree of one polynomial, the degree of other polynomial may increase. However, this algorithm has got important applications in the decoding of two dimensional codes, e.g.,2-D fire codes. Since the feedback polynomial set forms a Groebner basis, this algorithm may also be

used for constructing a basis for two dimensional polynomial ideals.

REFERENCES

1. Arcese A. and Reed I.S., "Linear prediction on a Galois field", Digital Signal Processing-84, Ed. Capellini V. and Constantinides A.G.S., Elsevier Science Publishers, B.V. North Holland, 1984.

2. Berger T., *Rate-distortion Theory: A Mathematical basis for data compression* ,Prentice Hall Inc.,Englewood-Cliffs, 1971.

3. Berlekamp E.R., *Algebraic coding theory* ,McGraw Hill, New York, 1968.

4. Blahut R.E., *Fast algorithms for digital signal processing*, Addison Wesley, Reading, Mass. 1985, Chapter 2.

5. Chaitin G.J., "On the length of programs for computing finite binary sequences", Journal of ACM, 13, 547-569, 1966.

6. Chaitin G.J., "Information-Theoretic computational complexity", IEEE Trans. on Information Theory", IT-20, 10-15, 1974.

7. de Luca A., "Complexity and information theory", *Coding and Complexity* , Ed. Longo G., 207-269, Springer Verlag, New York, 1975.

8. Fornasini E. and Marchesini G., " Structure and properties of two-dimensional systems", *Multidimensional Systems: Techniques and Applications*, Ed. Tzafestas S.G., 37-88, Marcel Dekker Inc., New York, 1986.

9. Gaines B.R., "On the complexity of causal models", IEEE Trans. SMC, Vol SMC-6,p56-59, 1976.

10. Gustavsson F.G., "Analysis of Berlekamp-Massey linear feedback shift register synthesis algorithm",IBM Journal of Research and Development, vol.20, no.3, pp.204-212, May 1976.

11. Hamming R.W., *Coding and Information theory*, Prentice Hall, Englewood-Cliffs,1986.

12.Imamura K. and Yoshide W., " A simple derivation of Berlekamp-Massey algorithm and some applications", IEEE Trans. Information Theory, IT-33, No.1, Jan 1987, pp. 146-150.

13. Jayant N.S. and Noll P., *Digital coding of waveforms and applications to speech and video*", Prentice Hall, Englewood-Cliffs, 1984.

14. Kalman R.E., Falb P.L. and Arbib M.A., *Topics in mathematical systems theory*, McGraw Hill, New York, 1961, Chapter 10.

15. Kalman R.E., On minimal partial realizations of a linear input output map", Aspects of network and system theory, Ed. Kalman R.E. and De Claris N, Holt, Rinlehart and Winston, New York, 1971.

16. Kalman R.E., " On partial realizations, transfer functions and canonical forms", Acta Polytechnica Scandinavia, Ma 31, Helsinki, 9-32, 1979.

17.Kung et. al.,"New results in 2-dimensional system theory, 2-D state space models, Part 2:Realizations and notions of controllability, observability and minimality", Proc. IEEE, Vol. 65,No.6,1977.

18. Mullick S.K., Jain P.A., Arun Kumar, "Partial realizations, Berlekamp-Massey algorithm and modelling of discrete time signals", To be published.

19. Nicolis J.S., *Dynamics of Hierarchical Systems: An evolutionary approach*, Springer-Verlag, 1986, pp178-181.

20. Sakata S., "On determinig the independent point set for doubly periodic arrays and encoding of two dimensional cyclic codes and their duals". IEEE Trans. Information Theory,IT-27, 556-565, 1981.

21. Sakata S., "Synthesis of 2-D LFSRs", Proc AAECC-5, Menorca, 1987, in Lecture notes in Computer Science, Vol. 356, Ed. L.Huguet and A. Poli, Springer, Berlin, 1989, 394-407.

22. Sakata S., "Finding a minimal set of linear recurring relations capable of generating a given finite two dimensional array", J. Symbolic Computation, 5, 321-377,1988.

23. Sakata S., "Extension of Berlekamp-Massey algorithm for N dimensions", Information and Computation, Vol. 84,No.2, 207-239,1990.

24. Ziv J. and Lempel A., "A universal algorithm for sequential data compression", IEEE Trans. on Information Theory, IT-23, no.3, pp.337-343, May 1979.

APPENDIX A

CORRECT PROOF FOR THE EXPECTED NUMBER OF STATES FOR GAINES MODEL

In section 1.2.2 it was pointed out that the proof by Gaines, given for deriving a bound on the expected number of states was wrong. The correct proof is given below. From equation (1.4) the average number of models for S-1 states is given by

$$\bar{M}(S-1) = \frac{1}{S-1} \sum_{R=1}^{S-1} R[R2^R-(R-1)2^{(R-1)}] \qquad \ldots(A.1)$$

Therfore the average number of S state models is obtained by simply replacing the S-1 in the above expression by S, and after some algebraic simplification , we get,

$$\bar{M}(S) = ((S-1) + \frac{2}{S} - \frac{2}{S2^S})/2^S \qquad \ldots(A.2)$$

If μ_S is the average number of states, then

$$\mu_S 2^S > \mu_S 2^{\mu_S} = \bar{M}(S) \qquad \ldots(A.3)$$

From (A.2) and (A.3) we get

$$\mu_S > S-2 \qquad \ldots(A.4)$$

From (1.2) and (A.4),

$$\mu_S > N - \log_2 N - 2 \qquad \ldots(A.5)$$

The ratio R_N of the expected number of states to the number of observations is given by

$$R_N = \frac{\mu_S}{N} > (N - \log_2 N - 2)/N \qquad \ldots(A.6)$$

and $R_N \to \infty$ as $N \to \infty$. This is the result obtained by Gaines though by a falty proof.

```
/* ************************************************************
This program gives a set of minimal polynomials              *
which satisfy the two dimensional linear recurrence          *
relation for a given two dimensional array.This              *
uses Sakata"s extension of Berlekamp Massey Algorithm        *
for the two dimensinal case.                                 *
************************************************************
*/
#include "infil"
main()
{
        /* DECLARATIONS */

        /* ************************************************************/
        int     *h, *f, *g, *f1 , *g1;
        /* h,f,g are the pointers for the newpolynomial,
                   the current polynomial set and the auxiliary
                   polynomial set respectively */
        /* f1, g1 provide the working space */
        struct x_y_z {
                int     x;
                int     y;
                int     *z;
        };
        struct x_y_z s[NPOL], t[NPOL], c[NPOL], r, p, q;
        /* s = degree set  of the current polynomial set
                   t = degree set of auxiliary polynomial set
                   c = max point set of excluded point set */



        struct x_y_z s1[NPOL], t1[NPOL], c1[NPOL];

        int     i, j, k, l;
        int     dq, dg;
        int     validity(), checksdelta();
        int     next(), order_structure();
        int     order_array(), adjacency();
        int     berlekamp();
        int     auxiliary();
        int     copy_pol(), copy_struct();

        /************************************************************

        f  = calloc(sizeof (int)*ARSZ * ARSZ * NPOL );
        g  = calloc(sizeof (int)*ARSZ * ARSZ * NPOL );
        h  = calloc(sizeof (int)*ARSZ * ARSZ );
        f1 = calloc(sizeof (int)*ARSZ * ARSZ * NPOL );
        g1 = calloc(sizeof (int)*ARSZ * ARSZ * NPOL );


        *f = 1;
        for (i = 0; i < 10 ; i++) {
```

```
                        s[i].z = f + ARSZ * ARSZ * i;
                        t[i].z = g + ARSZ * ARSZ * i;
                        c[i].z = g + ARSZ * ARSZ * i;
                }
                p.x = 0;
                p.y = 0;
                next(p);
                l = 0;
                point_count = 1;
                if (point_count < P_COUNT) {
                        scanf("%d", &a[point_count.x][point_count_y]);
                        int     iN = 0;
                        int     iV = 0;
                        for (j = 0; j <= 1; j++) {
                                dq[j] = validity(&s[j] , a, p);
                                if (dq[j] == 0) {
                                        sV[iV].x = s[j].x;
                                        sV[iV].y = s[j].y;
                                        iV++;
                                } else {
                                        if ((check = checksdelta(&s[j], g)) == 0
) {
                                                sN[iN]. x = s[j].x;
                                                sN[iN]. y = s[j].y;
                                                tN[iN].x = tN[iN].x;
                                                tN[iN].y = tN[iN].y;
                                                iN ++;
                                        } else {
                                                int     tmp1, tmp2;
                                                int     tmp = 0;
                                                for (tmp1 = 0; tmp1 < s[j].x; tm
p1++)
                                                        for (tmp2 = 0; tmp2 < s[
j].y ;
        tmp2++) {
                                                                g1[(1+tmp-1) * A
RSZ* ARSZ
        +s[j].x* ARSZ+ s[j].y] = f[(j-1)*ARSZ*ARSZ + s[j].x *ARSZ+s[j].y];
                                                                t1[1+tmp].x = s[
j].x;
                                                                t1[1+tmp].y = s[
j].y;
                                                                c1[1+tmp].x
        = p.x - s[j].x;
                                                                c1[1+tmp].y
        = p.y - s[j].y;
                                                                tmp++;
                                                        }
                                                }
                                        }
                                }
                        }
                }
                newl = order_structure(c1, 1 + tmp);
                adjacency(s1, c1 , newl);
                order_array(t1);
```

```
                                for (i = 0; i < newl ; i++) {
                                        for (j = 0; j < iV; j++) {
                                                if (s1[i].x == sV[j].x)
                                                        for (i1 = 0; i1 < s[i].x
 i1++)
                                                                for (j1 = 0; j1
s[i].y; j1++)
                                                                        *(s1[i].
+
        ARSZ * i1 + j1) = *(s[i].z + ARSZ * i1 + j1);
                                        }


                                        for (j = 0; j < iN; j++) {
                                                for (j = 0; j < iN; j++) {
                                                        if (s1[i].x == sN[j].x)
                                                                berlekamp(h,
        sN[j], sN[j].z, s[j], t[j]->z, p, q, dq[j]);
                                                }
                                        }


                                        /* otherwise if none of the auxiliary
        polynomials are
                needed to give the new polynomial then use the auxiliary
                procedure*/
                                        auxiliary(h, s1[i], s[i]);

                                }



                                /* copy f1 to f; s1 to s; g1 to g; t1 to t; c1 t
c; c1hat to chat; */

                                copy_pol(f1, f);
                                copy_struct(s1, s);
                                copy_pol(g1, g);
                                copy_struct(t1, t);
                                copy_struct(c1, c);
                                copy_struct(c1hat, chat);
                                l = newl;
                                next(p);

                        }
                }
                for (i = 0; i < l; i++) {
                        for (j = 0; j < s[i].x; j++)
                                for (k = 0; k < s[i]->y; k++)
                                        printf("%d", f[j][k]);
                        printf("\n");
                }
                free(f);
                free(g);
                free(h);
                free(f1);
                free(g1);
        }
```

```
#include "infil"
int     berlekamp(h, r, f, s, g, t, p, q, dq)
int     *h, *f, *q;
struct x_y_z {
        int     x;
        int     y;
        int     *z;
} *r, *s, *t, *p, *q;
int     dq;


{
        /*h= newpoly;deg(h)=r;*/
        /*f=poly to be modified deg(f)=s ord(f)=q-s*/
        /*g=max expoly   deg(g)=t ord(g)=p-t*/
        /*dq=delta*/
        struct x_y_z *l;
        int     i, j, k;
        int     *temp;
        int     copy(), z_mult(), sc_mult(), poly_sub();

        temp = malloc(sizeof (int)*ARSZ * ARSZ);
        (r)->x = max((s)->x, (t)->x + (q)->x + (p)->x);
        (r)->y = max((s)->y, (t)->y + (q)->y + (p)->y);
        (l)->x = (r)->x - (s)->x;
        (l)->y = (r)->y - (s)->y;
        copy(h, f, s);
        z_mult(h, s, l);
        copy(temp, g, t);
        sc_mult(temp, t, dq);
        (l)->x = (r)->x - (q)->x + (p)->x - (t)->x;
        (l)->y = (r)->y - (q)->y + (p)->y - (t)->y;
        z_mult(temp, t, l);
        poly_sub(h, r, temp);
        free(temp);
        return(0);
}
```

```
#include "infil"
int     auxiliary(h, r,  s, g)
int     *h, *f, *g;
struct x_y_z {
        int     x;
        int     y;
        int     *z;
} *r, *s ;



{
        /*h= newpoly;deg(h)=r;*/
        /*f=poly to be modified deg(f)=s ord(f)=q-s*/
        /*g=max expoly   deg(g)=t ord(g)=p-t*/
        /*dq=delta*/
        struct x_y_z *l;
        int     i, j, k;
        int     *temp;
        int     copy(), z_mult(), sc_mult(), poly_sub();

        temp = malloc(sizeof (int)*ARSZ * ARSZ);
        (l)->x = (r)->x - (s)->x;
        (l)->y = (r)->y - (s)->y;
        copy(h, f, s);
        z_mult(h, s, r);
        free(temp);
        return(0);
}
```

```
#include "infil"
int     copy(h, temp, p)
struct x_y_z {
        int     x;
        int     y;
        int     *z;
} *p;
int     *h, *temp;
{
        int     i, j;
        for (i = 0; i <= p->x; i++)
                for (j = 0; j <= p->y; j++)
                        *(h + (ARSZ * i) + j) = *(temp[ARSZ*i +j]);
        return(0);
}
```

```
#include "infil"
int     poly_sub(h, r, f);
int     *h, *f;
struct x_y_z {
        int     x;
        int     y;
        int     *z;
} *r;
{
        int     i, j;
        int     mod();
        for (i = 0; i <= r->x ; i++)
                for (j = 0; j <= r->y; j++)
                        *(h + ARSZ * i + j) =
                           mod( *(h + ARSZ * i + j) - (*(f + ARSZ * i
j)), M);
        return(0);
}
```

```
#include "infil"
int     z_mult(h, r, s)
struct x_y_z {
        int     x;
        int     y;
        int     *z;
} *r, *s;
int     *h;
{
        int     i, j;
        int     *temp;
        struct x_y_z *p;
        int     copy();

        temp = malloc(sizeof (int)*ARSZ * ARSZ);

        for (i = 0; i < s->x; i++)
                for (j = 0; j <= (r->y + s->y); j++)
                        *(temp + ARSZ * i + j) = 0;
        for (i = s->x; i <= (r->x + s->x); i++)
                for (j = 0; j < s->y; j++)
                        *(temp + ARSZ * i + j) = 0;
        for (i = s->x; i <= (s->x + r->x); i++)
                for (j = s->y; j <= (s->y + r->y); j++)
                        *(temp + ARSZ * i + j) = *(h + ARSZ * (i - s
+ j - s->y);
        copy(h, temp, r);
        free(temp);
        return(0);
}
```

```
#include "infil"
int     sc_mult(h, r, dq)
struct x_y_z {
        int     x;
        int     y;
        int     *z;
} *r;
int     *h;
int     dq;
{
        int     i, j;
        int     mod();
        for (i = 0; i <= r->x; i++)
                for (j = 0; j <= r->y; j++)
                        *(h + ARSZ * i + j) = mod((*(h + ARSZ * i + j)
* dq, M);
        return(0);
}
```

```
#include "anti.i"
int     copy(h, temp, p)
struct x_y_z {
        int     x;
        int     y;
        int     *z;
} *p;
int     *h, *temp;
{
        int     i, j;
        for (i = 0; i <= p->x; i++)
                for (j = 0; j <= p->y; j++)
                        *(h + (ARSZ * i) + j) = *(temp[ARSZ*i +j]);
        return(0);
}
```